Advanced Computer Programming

Lecture 5

Alberto Ceselli
alberto.ceselli@unimi.it

Dipartimento di Informatica
Università degli Studi di Milano

October, 31st, 2014

Programming in the large

- Design and analysis of algorithms: small, but complex, problems to be solved with compact programs
- Real settings: large problems, composed by many subproblems, involving many programmers ...
- Software Engineering aims at tackling the problems from an high-level and management perspective
- Programming must provide the techinical "operational" counterpart
- Main objectives: maintainability, reuse, portability ...
- Main features: *factoring*, *locality*
- Main principles: *abstraction* and *modularity*

Process abstraction

The simplest example of process abstraction is the subprogram:

sortInt(list, listLen)

it is an abstraction of a sorting process.

Data abstraction

The simplest example of data abstraction are the built-in types (e.g. floating point):

the programmer applies to floating point variables the operators of the programming language (+, / ...), disregarding which is their effect at a bit-level

as a side effect, this ensures code *portability*.

User–defined ADT

Definition (User–defined Abstract Data Types)

The ADT is a *data type* satisfying the following requirements:

- The declarations of the *type* and *operations* on the type elements are contained in a *single syntactic unit*
- The internal representation of the type elements is *hidden* to code units using that type, therefore, the *only* operations allowed on the elements of that type are those listed in the declaration.

Features of an ADT

- The ADT is described by an *interface*
- The *implementation* (definition) can be in the same compiling unit (e.g. source file) or in a different unit.
- External units must be able to use the defined type as well.

- **Information hiding:** the internal representation (implementation) are *irrelevant* to the rest of the program
- **Encapsulation:** the only operations available to code outside the definition unit are those declared in the interface

Example of a declaration (interface) of an ADT

Type: stack
Operations:

- create() $\rightarrow$ stack
- destroy(stack)
- empty(stack) $\rightarrow$ bool
- push(stack, element) $\rightarrow$ stack
- pop(stack) $\rightarrow$ stack
- top(stack) $\rightarrow$ element

Example of using an ADT

```
...
s = create();
s = push(s, 21);
s = push(s, 3);
if ( ! empty(s) ) write top(s);
...
```

Example of an implementation of an ADT

- stack $\rightarrow$ array and integer 'pointer' to the first free component
- create(): allocates the array and gives back the pointer to its first component
- destroy(stack): releases the array memory
- empty(stack): is the pointer null?
- ...

Example of an alternative implementation

- stack $\rightarrow$ linked list
- create(): returns a NULL pointer
- destroy(stack): frees all the elements in the list
- empty(stack): is the pointer NULL?
- push: creates a new elements and links it to the list

Remark: the 'client' code is not changing at all!
Remark: any subprogram can be implemented using a different paradigm (it is kind of "fundamental problem").